Obliczenia inżynierskie w chmurze

Michał Karpus, 307429

Porównanie algorytmów całkowania numerycznego – metody: trapezów, Simpsona i Monte Carlo

1. Cel i opis projektu

Celem projektu było zbadanie wydajności trzech algorytmów całkowania numerycznego: metod trapezów, Simpsona i Monte Carlo z wykorzystaniem technologii chmurowych. Do przeprowadzenia obliczeń napisano 4 programy w języku Python, uruchomiono maszynę wirtualną w serwisie Microsoft Azure, skopiowano pliki na maszynę wirtualną dzięki programowi FileZilla oraz stworzono obraz maszyny w aplikacji Docker.

2. Programy w języku Python

Na potrzeby projektu napisano 4 programy w języku Python:

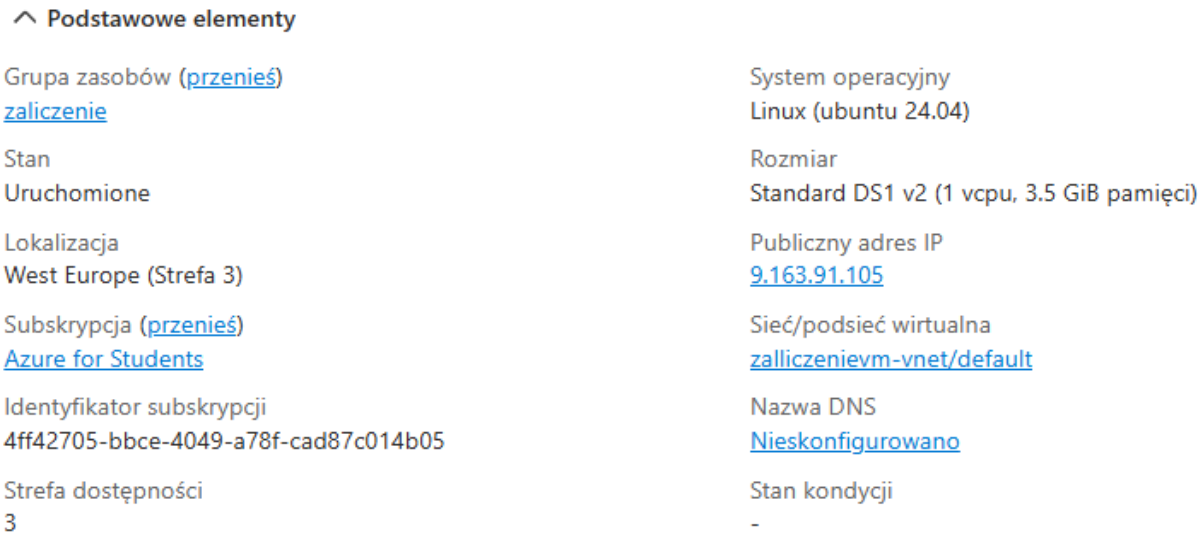
a) main.py – program główny,
b) trapezoidal_integration.py – program obliczający wartość całki oznaczonej przy pomocy metody trapezów,
c) simpson_integration.py – program wykorzystujący do obliczeń metodę Simpsona,
d) monte_carlo_integration.py – program wykorzystujący metodę Monte Carlo.

W programie main.py umieszczono funkcję *main*, która obsługuje wejście programu, przekształca wzór wpisany w linię komend na odpowiedni do dalszych obliczeń, przekazuje funkcję innym programom, a na koniec zwraca wynik w konsoli. Odpowiada także za wyświetlanie komunikatów o ewentualnych błędach.

Programy: trapezoidal_integration.py, simpson_integration.py i monte_carlo_integration.py składają się z pojedynczych funkcji realizujących obliczenia przy pomocy zadanego algorytmu oraz zwracają wartości całek oznaczonych wpisanych przez użytkownika. Wykorzystano biblioteki *sympy* i *random*. Listingi programów są dostępne w osobnych plikach w katalogu.

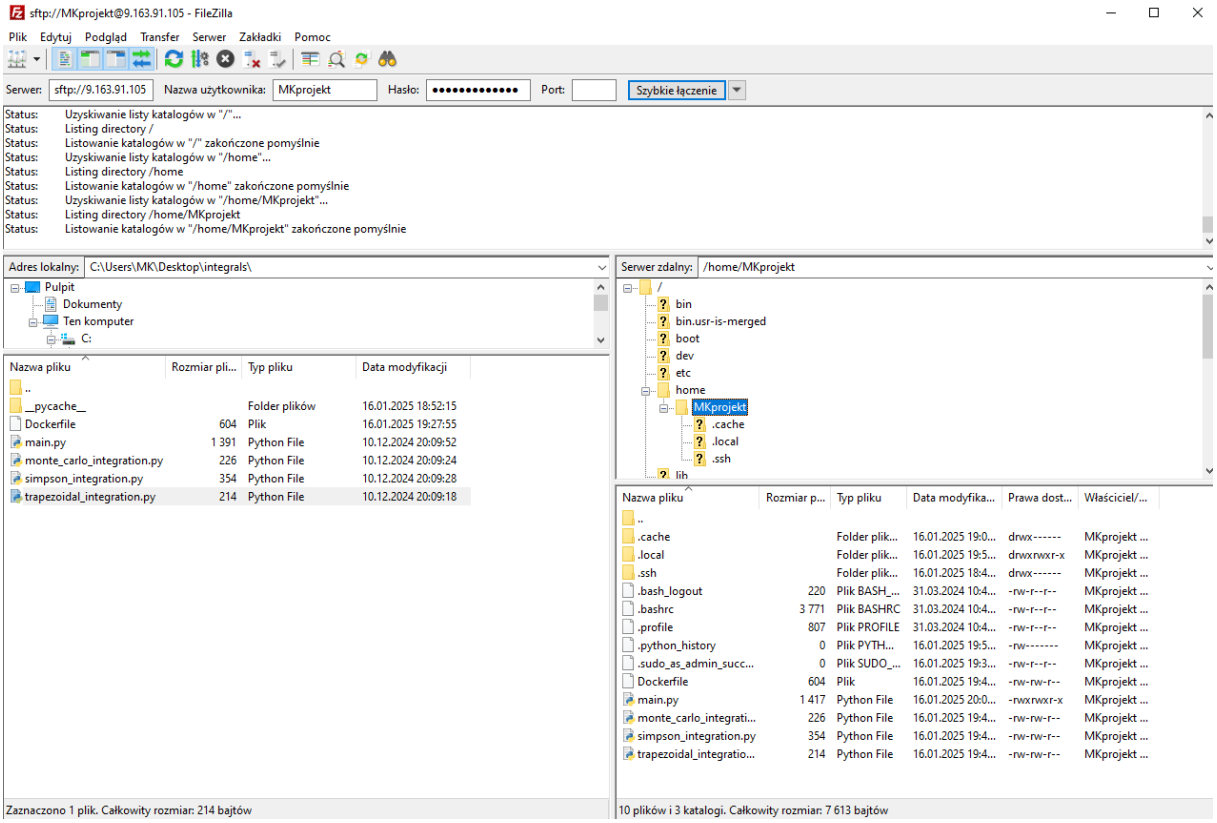
3. Schemat wykonania projektu

Projekt podzielono na mniejsze elementy wykorzystujące różne aplikacje i środowiska. Pracę rozpoczęto od napisania programów opisanych w rozdziale 2. Następnie uruchomiono maszynę wirtualną w serwisie Microsoft Azure. Posiadała ona 1 procesor wirtualny i 3,5 GiB pamięci do wykorzystania. Zainstalowano na niej system operacyjny Linux w wersji Ubuntu 24.04. Podstawowe informacje na temat tej maszyny przedstawiono na rysunku 1.



Rys. 1. Podstawowe parametry maszyny wirtualnej biorącej udział w projekcie.

Następnie skopiowano programy z rozdziału 2. wraz z plikiem Dockerfile na maszynę wirtualną. Zrobiono to przy pomocy aplikacji FileZilla. Na rysunku 2. Przedstawiono katalogi z poszczególnymi plikami na komputerze prywatnym i maszynie wirtualnej o nazwie *MKprojekt*.



Rys. 2. Widok aplikacji FileZilla.

Kolejnym krokiem było zalogowanie się do maszyny wirtualnej poprzez linię poleceń systemu Windows. Następnie zainstalowano na niej aplikację Docker i zbudowano obraz o

nazwie *python39*. W tym momencie można już było uruchomić kontener, który automatycznie rozpoczynał działanie programu *main.py*. Po zakończeniu obliczeń usuwano kontenery zbudowane w trakcie. Poniżej przedstawiono komendy, które wpisano w wiersz poleceń. Dwie ostatnie odpowiadają za usunięcie kontenera.

ssh nazwa@IP_maszyny

sudo apt-get update

sudo apt install docker.io

sudo docker build -t python39 .

sudo docker run –name python39 -it python39

sudo docker ps -a

sudo docker rm numer_kontenera

4. Wyniki obliczeń

Do przeprowadzenia obliczeń wykorzystano funkcję wielomianową

$$f(x) = x^3 + 4x^2 + 5x + 10$$

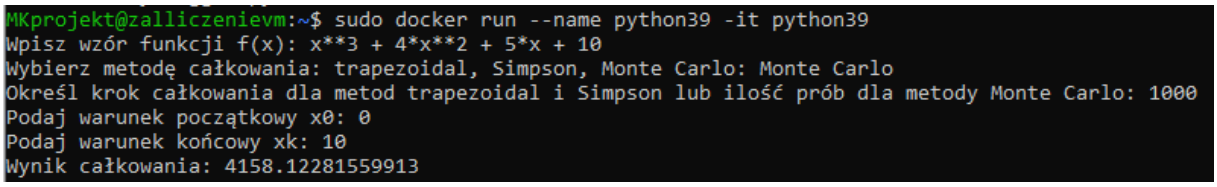
a granice całkowania ustalono na 0 i 10. Powyższa całka oznaczona jest równa $4183\frac{1}{3}$. Na rysunkach 3. – 5. przedstawiono wiersze poleceń po wykonaniu obliczeń z krokiem całkowania równym 0,01 i tysiącem (10 segmentów po 100) prób w metodzie Monte Carlo. W tym przypadku metoda Simpsona dała najdokładniejszy możliwy wynik, metoda trapezów pozwoliła uzyskać wynik z dokładnością do 2 miejsc po przecinku, zaś metoda Monte Carlo nie doprowadziła do satysfakcjonującego wyniku. Jest to spowodowane względnie małą ilością rozpatrywanych prób, co jest bardzo istotne w przypadku metod probabilistycznych. Programy wykonywały się w ułamkach sekund, bez odczuwalnej różnicy.

```
MKprojekt@zalliczenievm:~$ sudo docker run --name python39 -it python39
Wpisz wzór funkcji f(x): x**3 + 4*x**2 + 5*x + 10
Wybierz metodę całkowania: trapezoidal, Simpson, Monte Carlo: trapezoidal
Określ krok całkowania dla metod trapezoidal i Simpson lub ilość prób dla metody Monte Carlo: 0.01
Podaj warunek początkowy x0: 0
Podaj warunek końcowy xk: 10
Wynik całkowania: 4183.33650000000
```

Rys. 3. Wynik obliczeń metodą trapezów, krok całkowania 0,01.

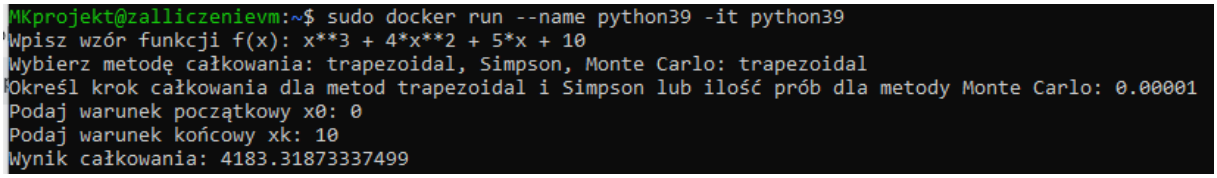
```
MKprojekt@zalliczenievm:~$ sudo docker run --name python39 -it python39
Wpisz wzór funkcji f(x): x**3 + 4*x**2 + 5*x + 10
Wybierz metodę całkowania: trapezoidal, Simpson, Monte Carlo: Simpson
Określ krok całkowania dla metod trapezoidal i Simpson lub ilość prób dla metody Monte Carlo: 0.01
Podaj warunek początkowy x0: 0
Podaj warunek końcowy xk: 10
Wynik całkowania: 4183.33333333333
```

Rys. 4. Wynik obliczeń metodą Simpsona, krok całkowania 0,01.

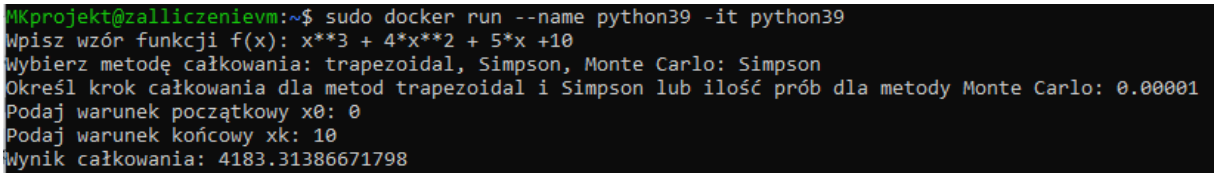


Rys. 5. Wynik obliczeń metodą Monte Carlo, 1000 prób.

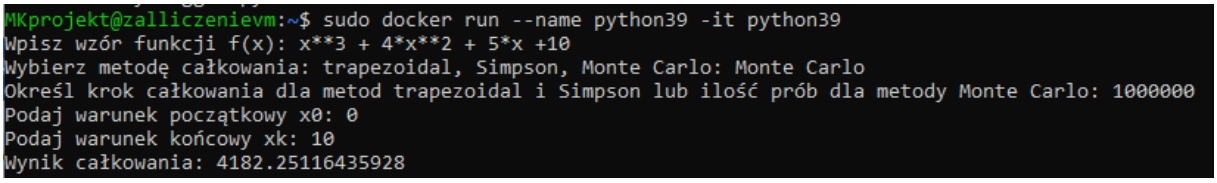
Przeprowadzono również obliczenia dla kroku 1000 razy mniejszego i 1000000 prób. Ich wyniki przedstawiono na rysunkach 6. – 8.



Rys.6. Wynik obliczeń metodą trapezów, krok całkowania 0,00001.



Rys.7. Wynik obliczeń metodą Simpsona, krok całkowania 0,00001.



Rys.8. Wynik obliczeń metodą Monte Carlo, 1000000 prób.

Powyższe obliczenia zajęły wyraźnie więcej czasu – odpowiednio ok. 11, 13 i 13 minut. Co ciekawe, żadne z nich nie dały satysfakcjonującego wyniku (tylko dokładność do 1. miejsca po przecinku lub wartość o ok. 1 mniejsza). Wskazuje to na obecność błędów numerycznych, co jest dość prawdopodobne przy takiej złożoności obliczeniowej.

5. Wnioski

Wszystkie badane algorytmy nadają się do numerycznego wyznaczania wartości całek oznaczonych funkcji wielomianowych. Metoda Simpsona pozwoliła uzyskać najdokładniejsze wyniki w krótkim czasie, zatem wskazane jest używanie jej w pierwszej kolejności przy rozważaniu takich funkcji, jak w projekcie. Metoda trapezów jest wystarczająca, ale minimalnie gorsza od metody Simpsona pod względem dokładności obliczeń. Metoda Monte Carlo pozwala na przeprowadzenie obliczeń, ale jest ona bardzo mała wydajna w przypadkach, jak badany – potrzebuje znacznie większego kosztu obliczeniowego. Można również uznać, że w przypadku takich funkcji, jak badana, nie należy całkować ze zbyt małym krokiem, gdyż prowadzi to w najlepszym przypadku do zużyciu większej ilości czasu.